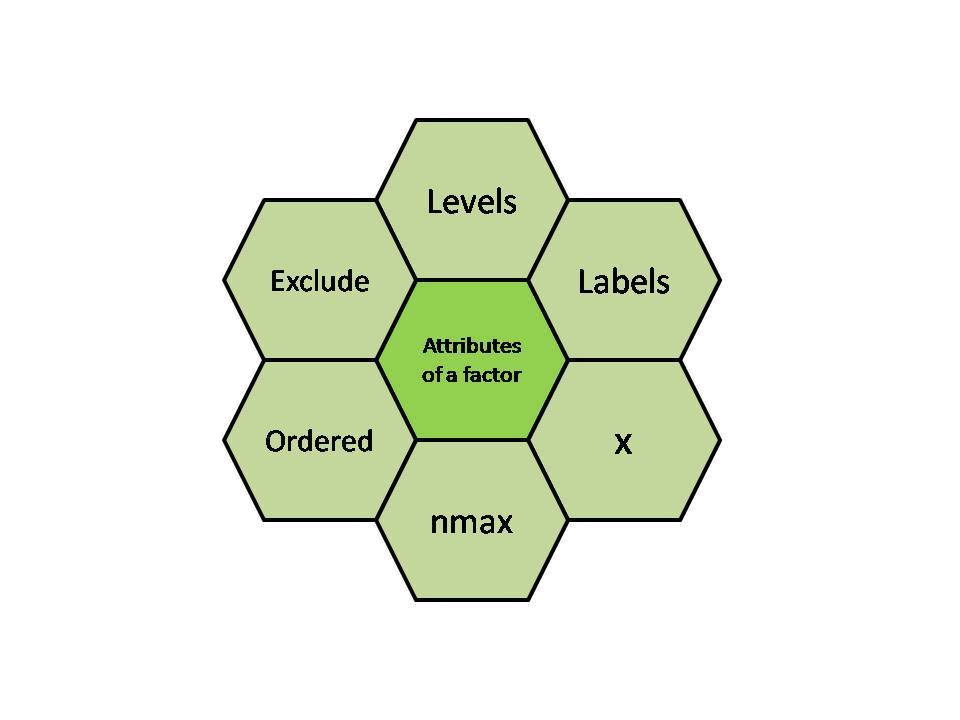
**Factors** in [R Programming Language](https://www.geeksforgeeks.org/r-programming-language-introduction/) are data structures that are implemented to categorize the data or represent categorical data and store it on multiple levels.

They can be stored as integers with a corresponding label to every unique integer. The R factors may look similar to character [vectors](https://www.geeksforgeeks.org/assigning-vectors-in-r-programming/), they are integers and care must be taken while using them as strings. The R factor accepts only a restricted number of distinct values. For example, a data field such as gender may contain values only from female, male, or transgender.

In the above example, all the possible cases are known beforehand and are predefined. These distinct values are known as levels. After a factor is created it only consists of levels that are by default sorted alphabetically.

**Attributes of Factors in R Language**

* x: It is the vector that needs to be converted into a factor.
* Levels: It is a set of distinct values which are given to the input vector x.
* Labels: It is a character vector corresponding to the number of labels.
* Exclude: This will mention all the values you want to exclude.
* Ordered: This logical attribute decides whether the levels are ordered.
* nmax: It will decide the upper limit for the maximum number of levels.

****

**Creating a Factor in R Programming Language**

The command used to create or modify a factor in R language is – factor() with a vector as input.   
The two steps to creating an R factor :

* Creating a vector
* Converting the vector created into a factor using function factor()

**Examples: Let us create a factor gender with levels female, male and transgender.**

# Creating a vector

x <-c("female", "male", "male", "female")

print(x)

# Converting the vector x into a factor named gender

gender <-factor(x)

print(gender)

Levels can also be predefined by the programmer.

# Creating a factor with levels defined by programmer

gender <- factor(c("female", "male", "male", "female"),

          levels = c("female", "transgender", "male"));

gender

Further one can check the levels of a factor by using function **levels()**.

**Checking for a Factor in R**

The function **is.factor()** is used to check whether the variable is a factor and returns “TRUE” if it is a factor.

gender <- factor(c("female", "male", "male", "female"));

print(is.factor(gender))

Function **class()** is also used to check whether the variable is a factor and if true returns “factor”.

gender <- factor(c("female", "male", "male", "female"));

class(gender)

**Accessing elements of a Factor in R**

Like we access elements of a vector, the same way we access the elements of a factor. If gender is a factor then gender[i] would mean accessing an ith element in the factor.

**Example**

gender <- factor(c("female", "male", "male", "female"));

gender[3]

More than one element can be accessed at a time.

**Example**

gender <- factor(c("female", "male", "male", "female"));

gender[c(2, 4)]

Subtract one element at a time.

**Example**

gender <- factor(c("female", "male", "male", "female"  ));

gender[-3]

* First, we create a factor vector **gender** with four elements: “female”, “male”, “male”, and “female”.
* Then, we use the square brackets **[-3]** to subset the vector and remove the third element, which is “male”.
* The output is the remaining elements of the **gender** vector, which are “female”, “male”, and “female”. The output also shows the levels of the factor, which are “female” and “male”.

**Modification of a Factor in R**

After a factor is formed, its components can be modified but the new values which need to be assigned must be at the predefined level.

**Example**

gender <- factor(c("female", "male", "male", "female"  ));

gender[2]<-"female"

gender

For selecting all the elements of the factor gender except ith element, gender[-i] should be used. So if you want to modify a factor and add value out of predefined levels, then first modify levels.

**Example**

gender <- factor(c("female", "male", "male", "female"  ));

# add new level

levels(gender) <- c(levels(gender), "other")

gender[3] <- "other"

gender

**Ordering Factor Levels**

Ordered factors levels are an extension of factors. It arranges the levels in increasing order. We use two functions: **factor()** and argument **ordered()**.

***Syntax:****factor(data, levels =c(“”), ordered =TRUE)*

***Parameter:***

* ***data:****input vector with explicitly defined values.*
* ***levels():****Mention the list of levels in c function.*
* ***ordered:****It is set true for enabling ordering.*

**Example:**

# creating size vector

size = c("small", "large", "large", "small",

         "medium", "large", "medium", "medium")

# converting to factor

size\_factor <- factor(size)

print(size\_factor)

# ordering the levels

ordered.size <- factor(size, levels = c(

  "small", "medium", "large"), ordered = **TRUE**)

print(ordered.size)

In the above code, the size vector is created using the c function. Then it is converted to a factor. And for the ordering factor, the () function is used along with the arguments described above. Thus the sizes are arranged in order.

The same can be done using the ordered function. An example of the same is shown below:

**Example:**

# creating vector size

size = c("small", "large", "large", "small",

         "medium", "large", "medium", "medium")

sizes <- ordered(c("small", "large", "large",

                   "small", "medium"))

# ordering the levels

sizes <- ordered(sizes, levels = c("small", "medium", "large"))

print(sizes)